A general diagrammatic algorithm for contraction and subsequent simplification of second-quantized expressions

Arteum D. Bochevarov and C. David Sherrill

Citation: J. Chem. Phys. **121**, 3374 (2004); doi: 10.1063/1.1774977
View online: http://dx.doi.org/10.1063/1.1774977
View Table of Contents: http://jcp.aip.org/resource/1/JCPSA6/v121/i8
Published by the American Institute of Physics.

A general diagrammatic algorithm for contraction and subsequent simplification of second-quantized expressions

Arteum D. Bochevarov[a] and C. David Sherrill[b]
School of Chemistry and Biochemistry, Georgia Institute of Technology, Atlanta, Georgia 30332-0400

(Received 3 May 2004; accepted 28 May 2004)

We present a general computer algorithm to contract an arbitrary number of second-quantized expressions and simplify the obtained analytical result. The functions that perform these operations are a part of the program Nostromo which facilitates the handling and analysis of the complicated mathematical formulas which are often encountered in modern quantum-chemical models. In contrast to existing codes of this kind, Nostromo is based solely on the Goldstone-diagrammatic representation of algebraic expressions in Fock space and has capabilities to work with operators as well as scalars. Each Goldstone diagram is internally represented by a line of text which is easy to interpret and transform. The calculation of matrix elements does not exploit Wick's theorem in a direct way, but uses diagrammatic techniques to produce only nonzero terms. The identification of equivalent expressions and their subsequent factorization in the final result is performed easily by analyzing the topological structure of the diagrammatic expressions. © *2004 American Institute of Physics.* [DOI: 10.1063/1.1774977]

I. INTRODUCTION

The rapid development and active usage of *ab initio* methods that include electron correlation have significantly changed the face of quantum chemistry that heretofore had been focusing mainly on the Hartree-Fock self-consistent field (SCF) theory and its semiempirical extensions. The molecular orbital theory provides the one-electron basis for the vast majority of the post-SCF approaches, most popular of which are various flavors of the many-body perturbation, configuration interaction, and coupled cluster theories. The correlation in such methods is introduced through excitations from a relatively small number of occupied to a large pool of virtual orbitals, and as the number of various matrix elements which must be evaluated in these approaches rises sharply with the maximum excitation level, their efficient implementation calls for a number of ingenious numerical and computational simplification techniques. Apart from the colossal number of computational operations needed to produce final numbers, there is another serious hindrance that currently bars researchers from performing high-accuracy quantum chemical calculations routinely. These days, the equations of quantum chemical models may contain a very large number of algebraic terms, so that expressions are difficult, if at all possible, to derive, analyze, and program. The idea to automate the generation and analysis of the mathematical expressions as well as produce computer code took real shape in the work of Janssen and Schaefer[1] (although it was by no means the very first among the works in which analytical calculations were partly performed by computer programs—see, for example, Ref. 2). It was followed by a number of programs which were, however, often aimed at some specific class of quantum chemical model, usually coupled cluster theory.[3–10] Recently a quite general program, the Tensor Contraction Engine (TCE) (Ref. 11) that became a part of a much larger and ambitious project[12] was developed by a number of researchers. A recent review by Hirata[13] describes the program's capabilities in detail. TCE, capable of handling, in principle, any algebraic theory, exploits Wick's theorem in order to derive the corresponding equations and then generates efficient computer code for the multiplication of the multidimensional arrays. However, TCE evaluates matrix elements that have a fixed form, which, although flexible enough to cover most of the currently used electronic structure theories, does not allow one to work with the matrix elements of arbitrary operators. Besides, TCE does not handle second-quantization operators explicitly, which is desirable in order to facilitate the construction of quantum chemical models which are not always written in terms of matrix elements.

In this paper we present a completely general algorithm to manipulate second-quantized expressions, whether scalars or operators. Our approach is based solely on the Goldstone diagrammatic technique,[14] popular among method developers. Diagrams, of which those of the Goldstone type are the most elementary and transparent, serve as a visual and topological (rather than algebraic) representation of second-quantized expressions, and proved to be extremely useful in many areas of quantum chemistry. Our algorithm can also be extended to treat Hugenholtz-type diagrams[15] which are essentially of the same type as Goldstone diagrams, but whose vertices absorb antisymmetrization. Hugenholtz diagrams might therefore be used to produce compact algebraic expressions that feature permutation operators, but Goldstone diagrams are advantageous when programmable expressions are needed or when the spin integration of the obtained expressions is planned. Also deserving consideration are the

[a]Electronic address: arteum.bochevarov@chemistry.gatech.edu
[b]Electronic address: sherrill@chemistry.gatech.edu

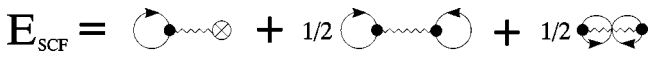

FIG. 1. The expression for the SCF energy in Goldstone diagrammatic notation.

spin-free diagrams of Kutzelnigg[16] which combine the assets of both Hugenholtz and Goldstone diagrams.

Several excellent introductory sources exist that describe in great detail the Goldstone diagrammatic technique[17–21] and it would seem excessive to outline it in this work. However, so many various types of diagrams emerged since their first usage in quantum chemistry (some of which look almost identical), that, for the benefit of the reader, we wish to briefly describe the notation we employ in drawing diagrams throughout this paper.

Each Goldstone diagram is a pictorial representation of the second-quantized expression for a given Fermi vacuum, so that each general index that belongs to the spin-orbital creation or annihilation operators is either a particle (p) or hole (h) state index. Particle (hole) states lie above (below) the Fermi vacuum. In this notation the particle (hole) creation operator is $a_p^\dagger$ (a_h), and the particle (hole) annihilation operator is a_p ($a_h^\dagger$). The diagrams we are using do not absorb numerical factors (which emanate from the definition of the second-quantized operators or from addition of one or more identical diagrams) to avoid confusion. They also do not include antisymmetrization, so that the expression for the closed shell SCF energy

$$E_{SCF}=\sum_h h_{hh}+\frac{1}{2}\sum_{hh'}(g_{hh',hh'}-g_{hh',h'h}) \tag{1.1}$$

will be represented diagrammatically as shown in Fig. 1. The first diagram (a ''bubble'') in it represents the core energy, the second (a ''double bubble'') represents the electron-electron Coulomb energy, and the third (an ''oyster'') represents the electron exchange energy.

The central idea behind the development of our program was to make it possible for the researcher (user) to perform any permissible operations on diagrams automatically. In other words, we would like to create a development tool that would be an aid in the manipulation and analysis of second-quantized expressions through which many quantum-chemical models can be written. In this paper we present only the algorithm of our diagrammatic approach and do not give many details about its specific implementation, since the algorithm may be realized in various computer languages. Our current version of Nostromo is written in the symbolic package MATHEMATICA (Ref. 22) and is being translated into the PYTHON langauge.

Nostromo works only with time-independent, discrete-basis, and particle conserving expressions. The requirement for the operators in the described algorithm to be particle conserving is not mandatory. With small and obvious modifications to the presented algorithm, the treatment of particle-non-conserving operators may also be included.

Diagrammatically, after the Fermi vacuum and the basis have been chosen, any second-quantized expression will be represented by one or more vertices and lines, coming in and out of these vertices. There will be vertices of four types only: particle creation and particle annihilation, particle creation and hole creation, particle annihilation and hole annihilation, and, finally, hole creation and hole annihilation. There will also be four types of lines—those that face down and are located below a vertex (so called down-below lines), and, using the analogous nomenclature, down-above, up-below, and up-above lines (see Fig. 2).

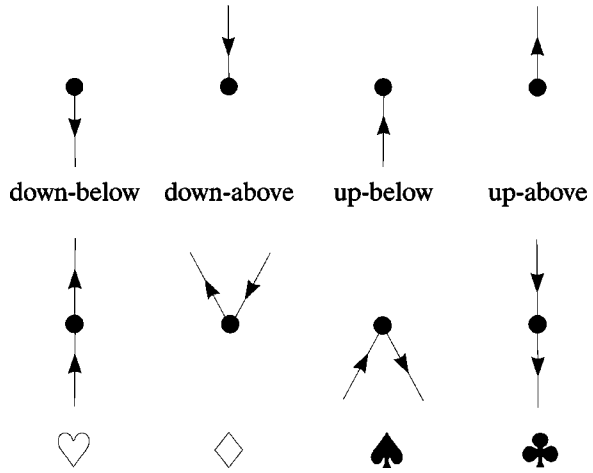

FIG. 2. Types of lines and vertices in Goldstone diagrams.

II. TEXTUAL REPRESENTATION OF GOLDSTONE DIAGRAMS

The unifying idea of Nostromo is that each diagram may be ''encoded'' by a text string, and vice versa, each such string of code may be translated back into a pictorial representation of the second-quantized expression—a Goldstone diagram. We call such a string of text a *textual diagram*. Every essential part of the diagram is represented by certain symbols in the text. In principle, this coding system does not provide a one-to-one correspondence between the pictures and algebraic formulas, but this is not a fundamental flaw and merely reflects the fact that there exists no one-to-one correspondence between algebraic expressions and diagrams. For example, an algebraic expression may be written as a diagram or as this diagram's mirror reflection, which do not always coincide. Conversely, a diagram that represents the multiplication of two vectors is translated back into the algebraic language equivocally—either as $\Sigma_i A_i B_i$ or as $\Sigma_i B_i A_i$. However, our coding system guarantees the isomorphic correspondence, so that any operation performed on the algebraic expression may be unequivocally conducted on its textual representation.

It should be stressed from the very start that diagrams (and, consequently, textual diagrams) are written only for normally ordered second-quantization expressions. Therefore, the analog of an n-particle operator in second quantization may be represented by Goldstone diagrams that have not only $2n$ open lines but also those that have $2n-2,2n-4,\ldots,0$ open lines. The reduction in the number of lines results from the contractions which appear as a ''penalty'' for writing the operator in normal form. The textual diagram contains the names of the operators that constitute this dia-

gram, as well as the information about how the different operators are connected by the particle and hole lines. Each text string starts with the name of an operator. It may be any operator which is included in this diagram. The name is always made of characters enclosed within two open square symbols (□). For example, □g□ means the name "g." The four types of vertices mentioned in the Introduction are denoted by the following four symbols, respectively: ♡, ♢, ♠, ♣ (see Fig. 2). After each such suit goes the information about the two lines that belong to the corresponding vertex. The line that leaves the vertex always goes first. It is necessary to show whether the line bears any index or if it is summed over. The line may be connected with another line, and it should be seen from the information in the text with which one. The information about each line is put between two delineators—an opening £ and a closing $. The line that faces up and which does not have any index (that is, summed over) is denoted by the symbol ↑. Similarly, the line that faces down and is summed over, is written as ↓. For example, the one-particle operator that includes all the particle creation and annihilation operators

$$\hat{h}=\sum_{pp'} h_{pp'}a_p^\dagger a_{p'} \tag{2.1}$$

is written in Nostromo in the following simple way: □h□♡£↑$£↑$. The one-particle operator in which the summation runs over all the indices—particle and hole,

$$\hat{h}=\sum_{pp'} h_{pp'}\{a_p^\dagger a_{p'}\}+\sum_{ph} h_{ph}\{a_p^\dagger a_h\}+\sum_{hp} h_{hp}\{a_h^\dagger a_p\}$$
$$+\sum_{hh'} h_{hh'}\{a_h^\dagger a_{h'}\}+\sum_{hh} h_{hh} \tag{2.2}$$

with the curly brackets indicating the normal ordering, is written in Nostromo as the sum

□h□♡£↑$£↑$+□h□♢£↑$£↓$+□h□♠£↓$£↑$
+□h□♣£↓$£↓$+□h□♣£~1$£~1$. (2.3)

The last term has a contraction and its structure in our coding system is described in the following paragraph. Further, suppose that we want to encode the operator

$$\hat{h}=h_{pp'}a_p^\dagger a_{p'}, \tag{2.4}$$

which has no summation over the indices. Then, in Nostromo coding system, we simply place the indices after the ↑ and ↓ symbols. Therefore, the above-mentioned operator will be written as □h□♡£↑$p$$£↑$p'$$.

The contraction of the line is denoted by the symbol ~followed by the contraction number, which replaces the arrow. For example, the contracted operator

$$\hat{h}_{Sc}=\left(\sum_{ij} h_{ij}a_i^\dagger a_j\right)_{Sc}=\sum_h h_{hh}, \tag{2.5}$$

where the subscript Sc denotes the scalar expression, is encoded as □h□♣£~1$£~1$. If the contraction retains the indices, as in

$$(h_{hh}a_h^\dagger a_h)_{Sc}=h_{hh}, \tag{2.6}$$

its code becomes □h□♣£~1$h$$£~1$h$$.

In case all the matrix elements $h_{ij\ldots}=\text{const}$, that is, the operator does not have a name, 1 is inserted between two open squares. For example,

$$\sum_{pp'} a_p^\dagger a_{p'} \tag{2.7}$$

is coded by □1□♡£↑$£↑$.

Using the described Nostromo coding system, it is possible to encode any complicated second-quantized expression. The linear combination of second-quantized expressions is written as a linear combination of textual diagrams.

III. NOSTROMO FUNCTIONS

The delineated encoding system bridges the gap between the pictorial representation of a second-quantized expression, which is appealing to a human, and the minimum amount of information needed for this second-quantized expression to be unequivocally recognized by a computer interpreter. It is straightforward now to write functions that work with diagrams by analyzing and parsing the textual diagrams.

Nostromo includes primitive functions, such as those that count the number of specific lines in a textual diagram; auxiliary functions that create textual diagrams (DiagramGenerator) or select them according to some criteria (for example, by the number of uncontracted lines, connectedness, etc.); and complicated, primary functions such as those that perform all the possible contractions among multiple diagrams (MultiContractor) and translate the received output (Translator) into the simplest algebraic expression, taking into account permutational and other types of symmetries. The automatic generation of diagrams occupied the attention of researchers in the past,[23,24] but these works were mostly concerned with fully contracted diagrams which appeared in perturbation expansions. DiagramGenerator produces all the topologically nonequivalent diagrams that constitute an n-particle operator, and in combination with the selection functions and MultiContractor, is capable of generating, in principle, any types of diagrams. While DiagramGenerator is easy to devise and implement, we nevertheless intend to characterize it in some detail in the following section, in order to render our paper more complete and our discussion of the subsequent material clearer. In comparison with DiagramGenerator, the functions MultiContractor and Translator have a rather complicated structure. MultiContractor is solely topological and does not directly employ Wick's theorem. The Translator function solves the important problem of identifying equivalent terms in the algebraic expression and their subsequent factorization. Instead of using a canonical form of any kind, it analyzes the topological structure of each diagram and equates terms with the same topological structure. The detailed description of these two functions will be given in Secs. V and VI.

IV. THE DIAGRAMGENERATOR FUNCTION

We begin with writing a general n-particle operator $Q^{(n)}$ (in which all n particles are equivalent) in second quantization form without explicitly specifying the Fermi level:

$$Q^{(n)}=\frac{1}{n!}\sum_{i_1 i_2,\ldots,i_n}\sum_{j_1 j_2,\ldots,j_n} Q_{i_1 i_2,\ldots,i_n,j_1 j_2,\ldots,j_n} \times a^{\dagger}_{i_1} a^{\dagger}_{i_2},\ldots,a^{\dagger}_{i_n} a_{j_n} a_{j_{n-1}},\ldots,a_1. \quad (4.1)$$

As we introduce the Fermi level, the indices $i_1 i_2,\ldots,i_n$ take on additional labels p (particle state) and h (hole state), and the operator $Q^{(n)}$ may be written as the sum of 2^{2n} terms, each of which is a summation analogous to Eq. (4.1) with the indices carrying distinct labels p or h. If we wish to work with $Q^{(n)}$ in a diagrammatic representation, we apply the Wick's theorem to it, so that it becomes the sum of the above-specified $2^{2n}=4^n$ terms, each brought to the normal order, plus a number of additional sums (also written in normal order) which contain one or more contractions—so called bubbles and oysters. It is easy to see that each of the uncontracted sums may be written diagrammatically as a series of n vertices positioned along a horizontal line. Each vertex can be any of ♡, ◇, ♠, ♣, totaling exactly 4^n different diagrams. As the vertices of a given diagram may be put in any order because of the particles' equivalence, the number of distinct diagrams may be reduced by bringing each of the 4^n uncontracted diagrams to some canonical form, for example, that in which all the vertices are arranged in the following order: ♡, ◇, ♠, ♣. A similar canonicalization is possible for the contracted terms. Concurrently to the described process, the DiagramGenerator function generates 4^n possible combinations of the four vertices for a n-particle operator, and produces a sum of the corresponding textual diagrams. Then, it adds to this sum all the textual diagrams with $1,2,\ldots,n$ contractions, thus forming a representation of the n-particle operator in normal form. After that, the vertices are rearranged according to the chosen canonical order. The process of generating diagrams, rearranging their vertices, and reducing the number of terms is illustrated for a two-particle operator $\hat{g}$ in Fig. 3.

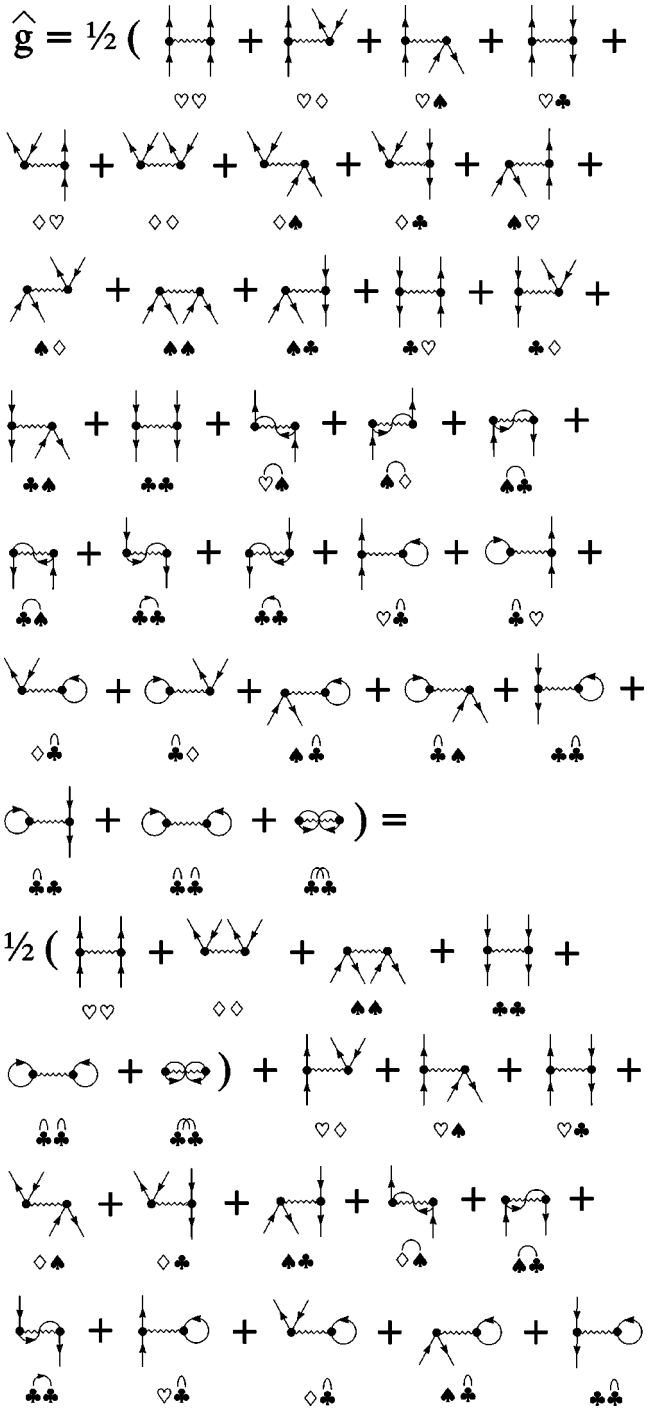

FIG. 3. The general two-particle operator in Goldstone diagrammatic notation: reducing the number of diagrams by identifying symmetry-equivalent ones.

V. THE MULTICONTRACTOR FUNCTION

As an input, MultiContractor receives a list of textual diagrams $\{A_1,A_2,\ldots,A_M\}$ as well as optional parameters that impose restrictions on the number or type of contractions to be performed. To give an example of the latter, we may mention a very often desired restriction that only fully contracted diagrams should be generated. An ordered sequence or list of (textual) diagrams $\{A_1,A_2,\ldots,A_M\}$ will be called an (uncontracted) configuration, with A_1 corresponding to the operator $\hat{A}_1$, A_2 corresponding to the operator $\hat{A}_2$, and so on. The order of diagrams in $\{A_1,A_2,\ldots,A_M\}$ is the same as the order of operators $\hat{A}_1,\hat{A}_2,..,\hat{A}_M$ in the product $\hat{A}_1\hat{A}_2\cdots\hat{A}_M$. If one or more contractions are performed among the diagrams of the sequence $\{A_1,A_2,\ldots,A_M\}$, the configuration is called a contracted configuration. At the outset, in order to make the explanations more clear, we shall assume that none of the above-mentioned restrictions are imposed and that all the possible contractions, conforming to a set of rules R, are allowed. MultiContractor realizes the following contraction rules R: $(R1)$ The diagrams to be contracted are positioned from above to below with the uppermost diagram corresponding to the leftmost textual diagram A_1 in the list and the lowermost diagram corresponding to the rightmost textual diagram A_M. $(R2)$ Up-lines are connected with up-lines and down-lines are connected with down-lines only. $(R3)$ Lines cannot be connected more than one time. $(R4)$ Any up- (down-) below line that belongs to the diagram A_k may be connected with any up- (down-) above line that belongs to the diagram A_l only if $k<l$. Any other connections are prohibited. Note that up-lines are con-

tracted independently from down-lines and that the set of all the possible contractions for a given configuration becomes a direct product of the set of up-contractions and down-contractions.

Suppose a configuration of M textual diagrams $\{A_1,A_2,\ldots,A_M\}$ is passed to the MultiContractor function together with the demand to generate, as a result of (possible) contractions, only those textual diagrams that have a specified number of up-below, up-above, down-below, and down-above lines. The number of performed contractions will be denoted as C. The following describes the MultiContractor algorithm.

(i) Calculate the number of up-above, up-below, down-above, and down-below lines (denoted as N_k^{ua}, N_k^{ub}, N_k^{da}, and N_k^{db}, respectively) for each diagram in $\{A_1,A_2,\ldots,A_k,\ldots,A_M\}$. Also calculate the total number of these lines. For example, the total number of up-above lines in the given configuration is $N^{ua}=\Sigma_{k=1}^{M}N_k^{ua}$.

(ii) In case MultiContractor is to perform the average value calculation, it need not generate diagrams with unconnected lines since they give zero contribution to the final result. The necessary condition for the list of diagrams to produce at least one fully contracted diagram is furnished by these equalities

$$N_1^{ua}=N_1^{da}=0,\quad N_M^{ub}=N_M^{db}=0, \tag{5.1}$$

$$\sum_{i=1}^{M-1}N_i^{ub}=\sum_{i=2}^{M}N_i^{ua},\quad \sum_{i=1}^{M-1}N_i^{db}=\sum_{i=2}^{M}N_i^{da}. \tag{5.2}$$

Requirements (5.1) are a direct consequence of the rule $(R4)$. If any of the statements (5.1) or (5.2) is not satisfied, MultiContractor instantly evaluates the contribution to the average value to be zero. Such a ''forecasting'' allows one to accelerate the calculation of scalar expressions significantly.

(iii) Calculate the maximal number of up-above, up-below, down-above, and down-below lines which can possibly be contracted (denoted as $C_{\max}^{ua}$, $C_{\max}^{ub}$, $C_{\max}^{da}$, and $C_{\max}^{db}$, respectively). Obviously, $C_{\max}^{ua}=C_{\max}^{ub}$ and $C_{\max}^{da}=C_{\max}^{db}$, therefore we shall only operate with $C_{\max}^{u}\equiv C_{\max}^{ua}$ and $C_{\max}^{d}\equiv C_{\max}^{da}$. Then $C^u=0,1,2,\ldots,C_{\max}^{u}$ and $C^d=0,1,2,\ldots,C_{\max}^{d}$.

Now we shall show how to calculate $C_{\max}^{u}$. The same arguments are applicable without any change to the calculation of $C_{\max}^{d}$. For a kth diagram, $1\leqslant k\leqslant M-1$, calculate $\gamma_k^u=\Sigma_{i=2}^{k}N_i^{ua}-\Sigma_{i=1}^{k-1}N_i^{ub}$. If $\gamma_k^u\geqslant 0$ (which means that the below-lines belonging to the diagrams with number l, $1\leqslant l\leqslant k-1$, may be fully contracted with the above-lines belonging to diagrams with number m, $2\leqslant m\leqslant k$), the number of the above-lines with which the below-lines of the diagram k may be contracted, is $Q_k^u=\min[\Sigma_{i=k+1}^{M}N_i^{ua},N_k^{ub}]$. If $\gamma_k^u<0$, $Q_k^u=\min[\gamma_k^u+\Sigma_{i=k+1}^{M}N_i^{ua},N_k^{ub}]$. Thus, Q_k^u is the maximal number of up-contractions for the kth diagram, given that all the below-lines belonging to the diagrams with numbers l, $1\leqslant l\leqslant k-1$, are maximally contracted. $C_{\max}^{u}$ is obtained by summing over all Q_k^u's,

$$C_{\max}^{u}=\sum_{k=1}^{M}Q_k^u. \tag{5.3}$$

In a similar manner,

$$C_{\max}^{d}=\sum_{k=1}^{M}Q_k^d. \tag{5.4}$$

(iv) Now we move on to building the *map* $\mathcal{M}$ of all the possible contractions, which is simply a data array containing the information of how the lines can be connected. Since up- and down-lines are connected independently, the map $\mathcal{M}$ is obtained by direct product of the up-map $\mathcal{M}^u$ and the down-map $\mathcal{M}^d$,

$$\mathcal{M}=\mathcal{M}^u\times\mathcal{M}^d. \tag{5.5}$$

The maps $\mathcal{M}^u=\{\mathcal{M}_j^u\}$ and $\mathcal{M}^d=\{\mathcal{M}_k^u\}$ are built in precisely the same way; therefore, we shall only expound on how one builds $\mathcal{M}^u$.

Suppose that each up-below line of the configuration $\{A_1,A_2,\ldots,A_M\}$ is given an index from 1 to N^{ub} (starting from the leftmost line of the uppermost diagram and ending at the rightmost line of the lowermost diagram). Then an array of N^{ub} elements, called an up-below *billet* and denoted B^{ub} will indicate the contraction state of the up-below lines. The analogous numbering may be performed on the up-above lines, and an up-above billet B^{ua} containing N^{ua} elements may also be constructed. If none of the up-below lines is contracted with the up-above lines, the billets B^{ub} and B^{ua} are filled with zeros. Should the kth up-below line be contracted with the lth up-above line, B^{ub} and B^{ua} will become

$$B^{ub}=\{0,0,\ldots,1,0,0,0,\ldots\}, \tag{5.6}$$

$$B^{ua}=\{0,0,0,\ldots,1,0,0,\ldots\}, \tag{5.7}$$

where 1 appears in B^{ub} in the kth position and in B^{ua} in the lth position. Similarly, should n up-contractions be performed, each of the billets B^{ub} and B^{ua} will contain n nonzero elements: $1,2,\ldots,n$ at the positions that indicate the positions of the contracted lines in the diagrams. The integers $1,2,\ldots,n$ serve as the dummy indices to show the structure of the contraction, and the combination of the billets B^{ub} and B^{ua} may represent an up-contraction. The list of all the legitimate up-contractions $\{[B^{ub},B^{ua}]_i\}$ may be brought to the form $\{B_i^{ub},\{B_{ij}^{ua}\}\}$ where every possible up-below billet B_i^{ub} is associated with the list of all the up-above billets, every one of which, if combined with B_i^{ub}, represents a valid up-contraction. The composite array that consists of elements $\mathcal{M}_i^u=[B_i^{ub},\{B_{ij}^{ua}\}]$ is the up-map $\mathcal{M}^u$.

(v) Since the up-map $\mathcal{M}^u$ includes all the up-contractions, its elements may be grouped into subarrays according to the number of connections between below and above lines. Thus, we shall have the subarray representing zero contractions ${}^{(0)}\mathcal{M}^u$ that consists of one element, as well as other subarrays ${}^{(1)}\mathcal{M}^u$, ${}^{(2)}\mathcal{M}^u$, ..., ${}^{(C_{\max}^u)}\mathcal{M}^u$.

Now we turn our attention to constructing the legitimate contractions $\{B_i^{ub},\{B_{ij}^{ua}\}\}$. Initially, we select the legitimate below-billets $\{B_i^{ub}\}$, and then, for each i, we select such above-billets $\{B_{ij}^{ua}\}$ that would form legitimate contractions, if combined with B_i^{ub}. For implementation purposes, it is convenient to build ${}^{(0)}\mathcal{M}^u$ first, then move on to ${}^{(1)}\mathcal{M}^u$, etc., and in the end, when all these submaps are generated,

TABLE I. The map $\mathcal{M}$ generated by MultiContractor when applied to the configuration $\{\Box h\Box \spadesuit \pounds\downarrow\$\pounds\uparrow\$,\Box f\Box \heartsuit \pounds\uparrow\$\pounds\uparrow \clubsuit \pounds\downarrow\$\pounds\downarrow\$,\Box g\Box \heartsuit \pounds\uparrow\$\pounds\uparrow\$ \diamond \pounds\uparrow\$\pounds\downarrow\$ \diamond \pounds\uparrow\$\pounds\downarrow\$\}$ with no restrictions on the number or type of contractions. The configuration and the billets are illustrated by Fig. 4.

	${}^{(k)}\mathcal{M}^{u}$		${}^{(k)}\mathcal{M}^{d}$	
Number of contractions	Below-billets	Above-billets	Below-billets	Above-billets
$k=0$	{0,0,0}	{0,0,0,0}	{0,0}	{0,0,0}
$k=1$	{1,0,0}	{{1,0,0,0},{0,1,0,0},{0,0,1,0},{0,0,0,1}}	{1,0}	{{1,0,0},{0,1,0},{0,0,1}}
	{0,1,0}	{{0,1,0,0},{0,0,1,0},{0,0,0,1}}	{0,1}	{{0,1,0},{0,0,1}}
$k=2$	{1,2,0}	{{1,2,0,0},{1,0,2,0},{1,0,0,2},{0,1,2,0},{0,1,0,2},{0,2,1,0},{0,2,0,1},{0,0,1,2},{0,0,2,1}}	{1,2}	{{1,2,0},{1,0,2},{0,1,2},{0,2,1}}

combine them into $\mathcal{M}^{u}$. There are a number of conditions (we call them below-conditions) that a below-billet must satisfy to be legitimate.

(a) It must have N^{ub} elements—zeroes and distinct integers from 1 to C^{u}.

(b) Below-lines that belong to the lowermost diagram should never be occupied, therefore the billet should have zeroes in the places that correspond to these lines.

(c) This condition puts restrictions on the number of contractions the below-lines of each operator may have. As all the particles are equivalent, all the below lines are treated on equal footing, and if any restrictions apply as to whether the lines of the given operator may be contracted, these restrictions should concern the total number of contractions and not the specific lines. For each diagram number k, the number of its below-lines which may be possibly contracted, in general, depends on whether the below-lines of the diagrams with numbers l, $1\leqslant l<k$, are already contracted. It may happen, for example, that some or all of the above-lines of the operators with numbers m, $k+1\leqslant m\leqslant M$, are already contracted with the below-lines of the diagrams with numbers l, $1\leqslant l<k$. The problem at this stage is that given only the above-billet we cannot know for sure *with which* above-lines the below-lines are contracted, we know only *which* below-lines are contracted. Therefore we select those below-billets which correspond to at least one legitimate contraction. For each of the diagrams (except the lowermost one) we check whether its below lines can possibly have as many connections as claimed for it in the given below-billet. The maximal number of the above-lines with which the below-lines of each diagram k may be connected (this number cannot exceed the number of connections indicated in the below-billet for the kth diagram if the billet is to be legitimate) is defined by the following scheme. Calculate $\lambda_k^u=\Sigma_{l=t+1}^{k}N_l^{ua}-\Sigma_{l=1}^{k-1}N_l^{ub}(\mathrm{conn})$, where t is the first diagram whose below-line is connected (the corresponding element in the billet is not zero) and $N_l^{ub}(\mathrm{conn})$ is the number of connected below-lines that belong to diagram number l (this information is extracted from the billet). If $\lambda_k^u\geqslant 0$ (which, similarly to the case of γ_k^u, means that the below-lines belonging to the diagrams with number l, $1\leqslant l\leqslant k-1$, are fully contracted with the above-lines belonging to diagrams with number m, $2\leqslant m\leqslant k$), then the maximal number of contractions for the below-lines of diagram number k is $P_k^u=\min[\Sigma_{i=k+1}^{M}N_i^{ua},N_k^{ub}]$, and, consequently, the number of uncontracted below-lines for diagram k cannot exceed $N_k^{ub}-P_k^u$. This is the sought-for condition. However, if $\lambda_k^u<0$, P_k^u is calculated according to the updated formula: $P_k^u=\min[\lambda_k^u+\Sigma_{i=k+1}^{M}N_i^{ua},N_k^{ub}]$.

(d) The billet should also comply with the optional restrictions on the number or type of contractions to be performed, which are given as input parameters at every MultiContractor function call.

(e) Finally, to avoid multiple counting of contractions, the nonzero integers in the below-billets are placed in the strictly ascending order.

All these conditions must be satisfied before the below-billet is accepted as valid. In an actual calculation, we generate all the billets that satisfy the criteria (a) and (e), and then select those that meet the criteria (b) and (c) and (d).

Once the below-billets $\{B_i^{ub}\}$ have been selected, it is easy to generate the above-billets $\{B_{ij}^{ua}\}$ for every i. The conditions for a legitimate above-billet are the following: (a) It must have N^{ua} elements—zeroes and distinct integers from 1 to C^{u}; (b) if contraction number k in the below-billet B_i^{ub} belongs to diagram number l then the contraction number l in all the above-billets $\{B_{ij}^{ua}\}$ should belong to the diagram number m, where $m>l$. This condition guarantees that no below-line is connected to an above-lying above-line. The fact that no restriction is laid down on the order of nonzero integers reflects the existence of permutations in many-body quantum theories. Examples of maps generated by MultiContractor are shown in Table I and illustrated by Fig. 4.

(vi) Given the configuration $\{A_1,A_2,\ldots,A_M\}$ and its map $\mathcal{M}$, it is now possible to generate all the contracted textual diagrams by converting the open line symbols $\uparrow$ and $\downarrow$, at the locations specified by $\mathcal{M}$, into the contraction symbols $\sim k$, where k is the contraction number. In case any of the diagrams in $\{A_1,A_2,\ldots,A_M\}$ come already partially or fully contracted with the maximal contraction number $k_{\max}$, the internal contractions numbers should be shifted by the $k_{\max}$, so that the dummy indices do not overlap. The textual diagrams with the substituted symbols $\uparrow$ and $\downarrow$ are concatenated to produce the final result. An example of the MultiContractor final result is given in Table II and illustrated by Fig. 5.

VI. THE TRANSLATOR FUNCTION

In this section, for simplicity's sake, we will be concerned with scalar expressions only, since it is clear how to

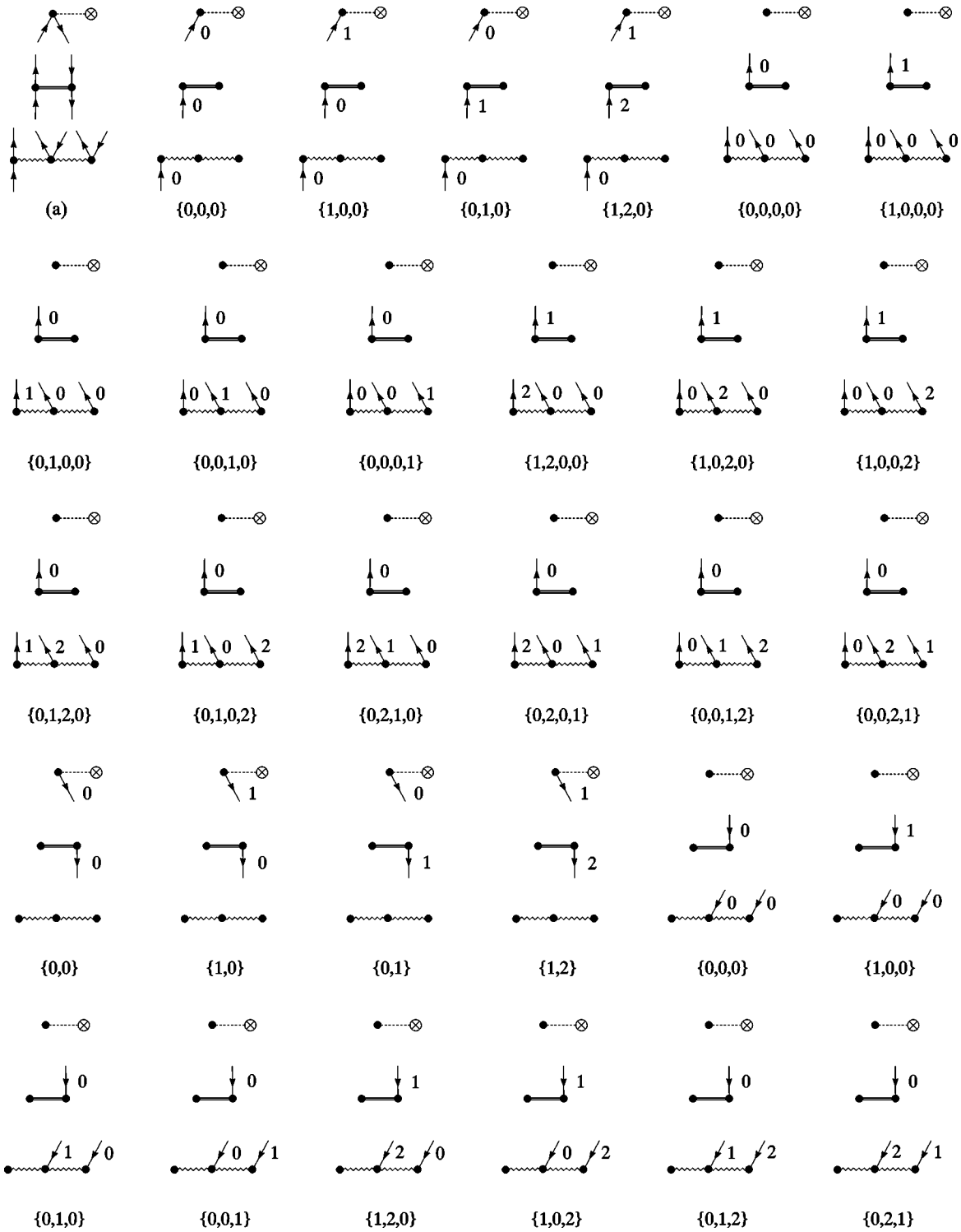

FIG. 4. The billets of the configuration (a) and the corresponding numbering of lines.

extend the shown results to the case of operators. As the batch of textual diagrams is output by MultiContractor, it is probable that some of these diagrams will produce the identical numerical result at arbitrary values of the elements of the corresponding multi-indexed arrays. We will call these diagrams (or algebraic expressions corresponding to them) equivalent. The equivalence may be brought forth by the different symbols for dummy indices (thus, the textual diagrams $\square h \square \clubsuit £\sim 1\$ £\sim 1\$$ and $\square h \square \clubsuit £\sim 2\$ £\sim 2\$$ and the corresponding algebraic expressions h_{ii} and h_{jj}, where the summation over the same indices is carried out, are equivalent), by permutational (akin to the well-known symmetries of the two-electron integrals in a physicist's notation $g_{ij,kl}=g_{ji,lk}$) or other symmetries of the operators (for example, $\square h \square \clubsuit £\sim i\$ £\sim j\$$ is equivalent to $\square h \square \clubsuit £\sim j\$ £\sim i\$$ if the matrix representing h is symmetric: $h_{ij}=h_{ji}$). As the number of contractions increases, the number of equivalent terms typically grows dramatically and it

TABLE II. Some of the textual diagrams produced as a result of applying MultiContractor to the configuration {□h□♠£↓$£↑$,□f□♡£↑$£↑♣£↓$£↓$,□g□♡£↑$£↑$◇£↑$£↓$◇£↑$£↓$} (the same as in Table I) with no restrictions on the number or type of contractions (the corresponding diagrams are shown in Fig. 5). The complete result includes 170 textual diagrams.

Total contractions	Textual diagrams
$k=0$	□h□♠£↓$£↑$□f□♡£↑$£↑♣£↓$£↓$□g□♡£↑$£↑$◇£↑$£↓$◇£↑$£↓$
$k=1$	□h□♠£~1$£↑$□f□♡£↑$£↑♣£
	~1$£↓$□g□♡£↑$£↑$◇£↑$£↓$◇£↑$£↓$
$k=2$	□h□♠£↓$£~2$□f□♡£~2$£↑♣£
	~1$£↓$□g□♡£↑$£↑$◇£↑$£~1$◇£↑$£↓$
$k=3$	□h□♠£↓$£~2$□f□♡£↑$£~3♣£
	~1$£↓$□g□♡£↑$£↑$◇£~3$£↓$◇£~2$£~1$
$k=4$	□h□♠£~1$£~3$□f□♡£↑$£~4♣£
	~2$£↓$□g□♡£↑$£↑$◇£~4$£~2$◇£~3$£~1$

would be highly desirable to develop a mechanism of their identification. A conventional approach to this problem is to postulate some canonical form for a fully contracted expression, convert all the terms to this canonical form, and simplify the result by means of cancellations, such as $a-a=0$, and additions, such as $a+a=2a$.

However, it may not be obvious how to lay down the rules for bringing the expression to a canonical form and in which order to execute them because some of these rules may be noncommutative. Furthermore, the problem is aggravated by the presence in the expression of two or more identical operators (the problem of so-called ''cyclic'' contractions), in which case the canonical form may not exist at all, and the canonicalization procedure requires additional operations, some of which scale factorially with the number of dummy indices.[13] Nostromo's Translator function is built on a different approach. It abandons the canonical form altogether and uses the topological structure of the diagrams to decide whether the two given terms are equivalent or not. The transparency and elegance of the diagrammatic simplification of expressions emphasizes the benefits of the use of diagrams in the automatization of second-quantized algorithms. We will demonstrate how our simplification procedure works for scalars (fully contracted diagrams), but it is straightforward to extend this technique to operators, which may contain terms with uncontracted lines. It may be easily seen why it is advantageous to use a diagrammatic representation, as opposed to an algebraic representation, in tackling the present problem. First, diagrams do not bear dummy indices, which exempts us from the necessity of their reshuffling and renaming. This way, any two expressions that differ by the names of dummy indices only will have the same diagrams. Second, by the virtue of the equivalence of all the electrons (or other particles, should the operators be written for them), the location of a particular vertex that belongs to a certain operator is immaterial—it is important only with vertices of which operators, and by means of which lines, this particular vertex is connected. Here we introduce the notion of cycle path as the recording of the journey [the vertices of which operators and by means of which lines (up, down) are encountered] that one makes if one sets off from a particular vertex and finally arrives at it again, forming a cycle. For example, the two equivalent algebraic expressions $g_{ij,ab}g_{ab,ij}$ and $g_{ij,ab}g_{ba,ji}$ that differ by a permutation will have different diagrams (shown in Fig. 6), but yet the same cycle paths. Third, cycles change only their direction but not their structure under the symmetry transformation of the type $h_{i,j}=h_{j,i}$ or $g_{ij,kl}=g_{il,kj}$, and so the backwards-directed cycle paths have to be generated as well, if this type of symmetry is to be taken into account. These observations allow us to conclude that the identity of a fully contracted mathematical expression is maintained by the number and topological structure of the cycles of its diagrammatic representation. Unfortunately, the present approach does not allow

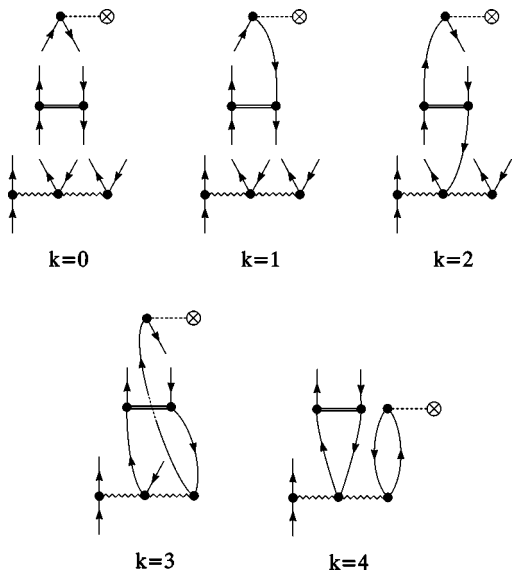

FIG. 5. The diagrams matching the textual diagrams from Table II, where k is the number of contractions.

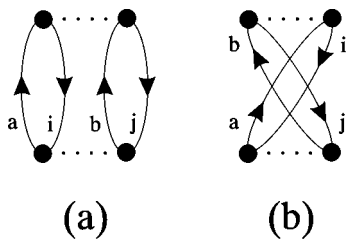

FIG. 6. The diagrammatic representation of equivalent algebraic terms $g_{ij,ab}g_{ab,ij}$ (a) and $g_{ij,ab}g_{ba,ji}$ (b).

one to handle in the general case special types of symmetry, such as the antisymmetry properties of coupled-cluster amplitudes $t_{ij,ab}=-t_{ij,ba}$, but such special symmetries may be efficiently treated by conventional canonization after the topological analysis has produced the maximally simplified (with respect to the above-mentioned symmetries) expression. The algorithm, from the point when a number of textual diagrams is received from MultiContractor, to the final, maximally simplified algebraic expression, proceeds in the following manner.

(a) The first textual diagram is taken and its topological signature (TS) is produced. The TS is an ordered array of the cycle paths of the selected diagram. Much of the advanced algebraic analysis of the topological structure of diagrams has been done in Ref. 25 (see also references therein) but for our simple purposes we shall not need it in this work. The way of recording cycle paths is to a certain degree optional. We adopted the following convention: the name of the operator from whose vertex (vertex 1) the cycle starts (say, h) is paired up with the index of the line (say, i) that connects it with the subsequent vertex (vertex 2), so that the pair becomes $\{h,i\}$. If the line does not have any index, but is a summation line instead (corresponding to some dummy index in algebraic notation), its ''index'' is written as $\uparrow$ or $\downarrow$, depending on whether the line faces up or down. The same operations are carried out for the vertex 2 and the line that leaves it, producing the second pair. The process goes on until the cycle ends at vertex 1. Then, all the *cyclic* permutations are generated from the obtained pairings (this operation scales only as N where N is the length of the cycle) and the permutation that is more ordered than the rest (order may be chosen arbitrarily), is selected as the cycle path. The cyclic permutations are needed to identify equivalent cycles, in case we started the paths from different vertices. Some examples of the cycles and their paths are presented in Fig. 7. In more complicated diagrams containing multiple cycles and identical operators, the structure of the TS may be more complex because it would be necessary to distinguish between the connectivities of the identical operators. When all the cycle paths are thus recorded, they are combined in an ordered way to produce the TS. The TS for the first term is saved along with the index 1 in the TS database.

(b) The second textual diagram is taken and its TS is generated.[26] If it is the same as the one already saved, term number 2 is declared equal to term number 1. If the TS of the second term is different from that of the first, it is saved along with number 2 in the TS database. The TS of the third term is compared with all the saved TS's and if does not match any, it is also saved. These operations continue until all the terms are processed, so that the final expression contains the minimal number of identical terms.

(c) If any special symmetry within some operators is present, the expression is further simplified by ordering indices that belong to this operator.

(d) The sign of the diagram is computed as $(-1)^{N_s+N_h}$ where N_s is the number of cycles (which is of course the number of cycle paths in TS) and N_h is the number of hole lines in the diagram.

(e) At this stage only the terms with different TS's exist.

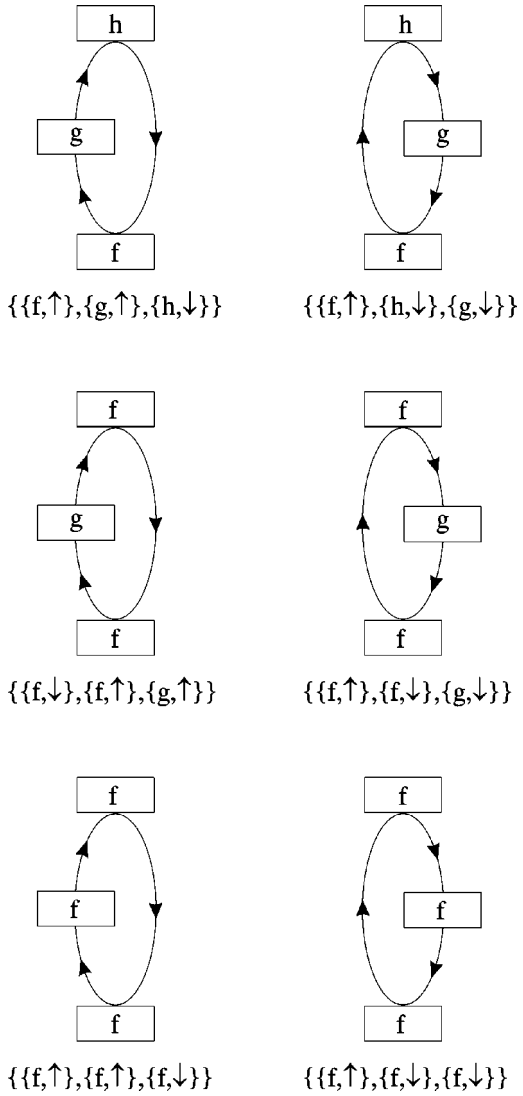

FIG. 7. The cycles and their topological signatures. Note that the cycles distinguished by direction only have different topological signatures, and that the topological signatures of the cycles with several identical operators are well defined.

It may happen, however, that there will be terms that have identical cycle paths. This means that the group of these terms may be further simplified by factorization, with the common cycle (or several common cycles) factored out.

(f) The resulting textual diagrams are converted into traditional algebraic notation, and the final result is produced.

VII. CONCLUSION

In this work we presented a general computer algorithm to manipulate, contract, and simplify second-quantized expressions, implemented in the prototype code Nostromo. Our approach is topological rather than algebraic and is based on the Goldstone diagrammatic representation of the second-quantized expressions. We showed that using diagrammatic techniques to automate the evaluation of the second-quantized expressions has a number of significant advantages over the traditional approaches which employ strings of creation and annihilation operators and subsequent application of Wick's theorem: first, representing diagrams with text

strings permits easy and intuitive manipulation of symbolic expressions on a computer; second, during the contraction process, the efficient generation of terms with the required structure only (such as fully contracted terms) is possible; and third, the topological analysis of the diagrams greatly facilitates the simplification of the output results.

Although we implemented the described algorithms in Nostromo, some technical programming work, not directly related to the mathematical and algorithmic issues discussed in this paper, still remains to be done before the generation and simplification of complicated expressions (such as coupled-cluster theory through quadruple excitations, or CCSDTQ) can be fully automated. Additionally, some customization of factorization schemes is necessary to bring the generated equations to their conventional form. Our paper discusses the factorization of diagrammatic expressions with identical cycles, but other cases where factorization is possible may arise, and, in general, these factorization schemes are not unique. A simple example is an expression for the matrix element $\langle\Phi_0|\hat{H}\hat{T}_1^2|\Phi_0\rangle$ that appears in the derivation of the CCSD equations,

$$\langle\Phi_0|\hat{H}\hat{T}_1^2|\Phi_0\rangle = \frac{1}{4}\sum_{ijab}\ (g_{ab,ij}t_i^a t_j^b - g_{ab,ij}t_j^a t_i^b). \tag{7.1}$$

This expression may be factorized in the traditional way,

$$\langle\Phi_0|\hat{H}\hat{T}_1^2|\Phi_0\rangle = \frac{1}{4}\sum_{ijab}\ (g_{ab,ij} - g_{ab,ji})t_i^a t_j^b$$

$$\equiv \frac{1}{4}\sum_{ijab}\ \langle ab||ij\rangle t_i^a t_j^b, \tag{7.2}$$

or in a less standard way,

$$\langle\Phi_0|\hat{H}\hat{T}_1^2|\Phi_0\rangle = \frac{1}{4}\sum_{ijab}\ g_{ab,ij}(t_i^a t_j^b - t_j^a t_i^b). \tag{7.3}$$

However, apart from these technical issues (which will be resolved in the PYTHON version of our code), the calculation and the simplification of the CCSDTQ matrix elements using the algorithms described in the paper is very fast—for example, the current version of Nostromo written in MATHEMATICA takes just minutes to generate the simplified symbolical expression for $\langle\Phi_{ijkl}^{abcd}|\hat{H}\hat{T}_4|\Phi_0\rangle$ on a personal computer.

The plans for future work include adding flexibility to deriving and manipulating symbolic expressions by extending the functionality of the present code. In particular, we are working on functions that will perform automatic spin integration and visualization of textual diagrams. The presented approach may also be geared towards other diagrams, for example, of Hugenholtz type. We believe that Nostromo is a first step in the direction of creating a much larger and more functional computer package for automatization of symbolic operations in quantum chemistry.

ACKNOWLEDGMENTS

A.D.B. is indebted to Alex Semenyaka and Dr. A. V. Zaitsevsky for introducing him to diagrammatic methods in quantum chemistry. Helpful discussions with Professor M. Nooijen and Dr. E. F. Valeev are also greatly appreciated. C.D.S. is a Blanchard Assistant Professor of Chemistry at Georgia Tech and acknowledges a National Science Foundation CAREER Award (Grant No. CHE-0094088) and a Camille and Henry Dreyfus New Faculty Award. The Center for Computational Molecular Science and Technology is funded through a Shared University Research (SUR) grant from IBM and by Georgia Tech.

[1] C. L. Janssen and H. F. Schaefer, III, Theor. Chim. Acta **79**, 1 (1991).
[2] G. D. Purvis III and R. J. Bartlett, J. Chem. Phys. **76**, 1910 (1982); G. D. Purvis and C. Nehrkorn, KOMMUTE program, QCPE, Bloomington, Indiana.
[3] S. A. Kucharski and R. J. Bartlett, Theor. Chim. Acta **80**, 387 (1991).
[4] X. Li and J. Paldus, J. Chem. Phys. **101**, 8812 (1994).
[5] M. Gotoh, K. Mori, and R. Itoh, Int. J. Quantum Chem. **56**, 163 (1995).
[6] T. D. Crawford, T. J. Lee, and H. F. Schaefer III, J. Chem. Phys. **107**, 7943 (1997); T. D. Crawford, Ph.D. thesis, University of Georgia, 1996.
[7] F. I. Harris, Int. J. Quantum Chem. **75**, 593 (1999).
[8] S. Hirata and R. J. Bartlett, Chem. Phys. Lett. **321**, 216 (2000).
[9] M. Nooijen and V. Lotrich, J. Chem. Phys. **113**, 494 (2000); M. Nooijen and V. Lotrich, *ibid.* **113**, 4549 (2000); M. Nooijen and V. Lotrich, J. Mol. Struct.: THEOCHEM **547**, 253 (2001).
[10] E. F. Valeev, SEQUANT, Version 1.0 (2003) (http://www.ccmst.gatech.edu/evaleev/).
[11] S. Hirata, *Tensor Contraction Engine: A Symbolic Manipulation Program for Quantum-Mechanical Many-Electron Theories*, version 1.0 (Pacific Northwest National Laboratory, Richland, WA, 2002).
[12] http://www.cis.ohio-state.edu/gb/TCE/
[13] S. Hirata, J. Phys. Chem. A **107**, 9887 (2003).
[14] J. Goldstone, Proc. R. Soc. London **239**, 267 (1957).
[15] N. M. Hugenholtz, Physica **23**, 481 (1957).
[16] W. Kutzelnigg, J. Chem. Phys. **82**, 4166 (1985).
[17] I. Lindgren, J. Morrison, *Atomic Many-Body Theory* (Springer, New York, 1982).
[18] S. Wilson, *Electron Correlation in Molecules* (Clarendon, Oxford, 1984).
[19] F. E. Harris, H. J. Monkhorst, and D. L. Freeman, *Algebraic and Diagrammatic Methods in Many-Fermion Theory* (Oxford University Press, New York, 1992).
[20] A. V. Zaitsevsky, *Methods of Many-Body Theory in Quantum Chemistry* (Moscow University Press, Moscow, 1993) (in Russian).
[21] T. D. Crawford and H. F. Schaefer III, in *Reviews in Computational Chemistry*, edited by K. B. Lipkowitz and D. B. Boyd (Wiley, New York, 2000), Vol. 14.
[22] Wolfram Research, Inc., MATHEMATICA, Version 5.0, Champaign, IL, 2003.
[23] J. Paldus and H. C. Wong, Comput. Phys. Commun. **6**, 1 (1973); H. C. Wong and J. Paldus, *ibid.* **6**, 9 (1973).
[24] U. Kaldor, J. Comput. Phys. **20**, 432 (1976).
[25] V. Kvasnička, Int. J. Quantum Chem. **21**, 1003 (1982).
[26] For every cycle in which all of its operators are symmetric, the forward- and backward-directed cycle paths should be generated. Thus, the diagram will be represented by N^2 topological signatures (where N is the number of such cycles) differing by at least one cycle direction. During the comparison with the topological signatures in the TS database, only one of these N^2 topological signatures should match an already saved one for the two diagrams to be declared equivalent. If there is no match, any of the N^2 topological signatures may be saved in the TS database to represent the given diagram.